

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-126161

(43)Date of publication of application : 11.05.1999

(51)Int.Cl.

G06F 9/22

G06F 9/22

(21)Application number : 09-290758

(71)Applicant : HITACHI LTD
HITACHI INFORMATION
TECHNOLOGY CO LTD

(22)Date of filing : 23.10.1997

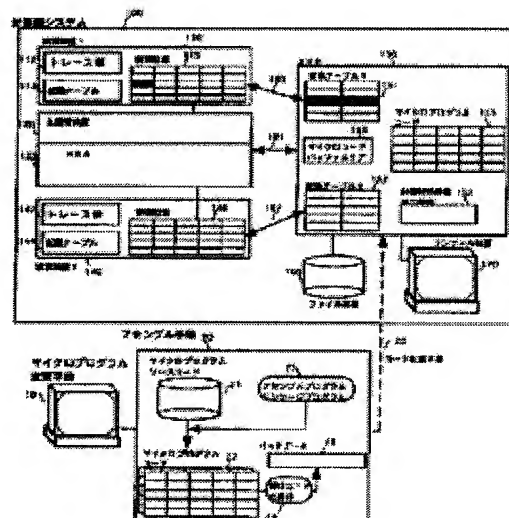
(72)Inventor : TAKESHIMA SEISUKE

(54) FAULT AVOIDING METHOD FOR CONTROL MEMORY

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a fault avoiding method for control memory that facilitates the maintenance management of a microprogram.

SOLUTION: The correspondence of an address (physical address) on a control memory 115 and an address (logical address) owned by the source code of the microprogram is added to a microprogram code 22 for loading to the control memory 115. At the time of initial microprogram load, an SVP tests the control memory and generates the correspondence table (translation table) of the physical address and the logical address so as to avoid the defective part of that memory, and based on the correspondence table, the physical address of the microprogram code 22 is rearranged so as to execute the initial load. The generated correspondence table is maintained as it is.



(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平11-126161

(43)公開日 平成11年(1999) 5月11日

(51)Int.Cl.⁶

G 0 6 F 9/22

識別記号

3 8 0

3 1 0

F I

G 0 6 F 9/22

3 8 0 B

3 1 0 D

審査請求 未請求 請求項の数4 O L (全 18 頁)

(21)出願番号 特願平9-290758

(22)出願日 平成9年(1997)10月23日

(71)出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(71)出願人 000153454

株式会社日立インフォメーションテクノロジー

神奈川県秦野市堀山下1番地

(72)発明者 竹島 増祐

神奈川県秦野市堀山下1番地 株式会社日

立インフォメーションテクノロジー内

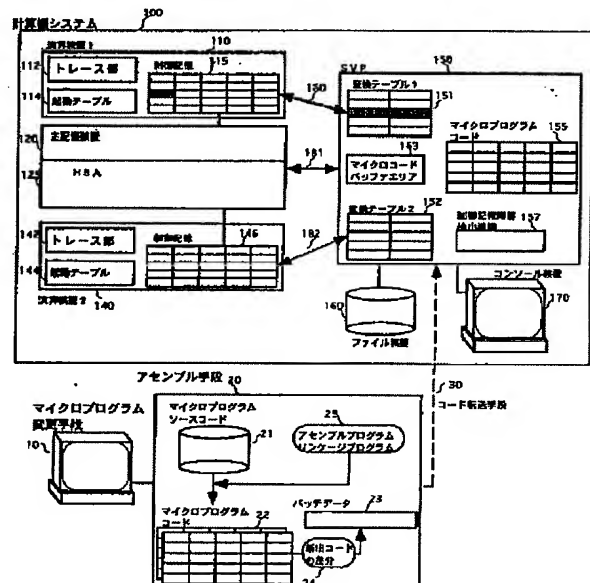
(74)代理人 弁理士 鈴木 誠

(54)【発明の名称】 制御記憶の障害回避方法

(57)【要約】

【課題】 マイクロプログラムの保守管理を容易にする制御記憶の障害回避方法を提供する。

【解決手段】 制御記憶にロードするためのマイクロプログラムコードに、制御記憶上のアドレス（物理アドレス）とマイクロプログラムのソースコードが持つアドレス（論理アドレス）の対応を付加する。SVPは、マイクロプログラム初期ロード時、制御記憶をテストし、その不良部位を回避するように物理アドレスと論理アドレスの対応表（変換テーブル）を生成し、該対応表にもとづいて、マイクロプログラムコードの物理アドレスを再配置し、初期ロードを実行する。生成した対応表はそのまま保守しておく。



【特許請求の範囲】

【請求項1】 マイクロプログラムが格納された制御記憶を内蔵する一つあるいは複数の演算装置と、前記演算装置の保守を司る保守制御装置とを具備してなる計算機システムにおいて、

制御記憶にロードするためのマイクロプログラムコードに、制御記憶上のアドレス（以下、物理アドレスと称す）とマイクロプログラムのソースコードが持つアドレス（以下、論理アドレスと称す）の対応を付加して保持し、

保守制御装置は、マイクロプログラムを制御記憶に初期ロードする際、制御記憶をテストし、その不良部位を回避するように物理アドレスと論理アドレスとの対応表を生成し、該対応表に基づいて、前記マイクロプログラムソースコードに付加された物理アドレスを再配置し、該再配置後の物理アドレスに従ってマイクロプログラムを制御記憶にロードすることを特徴とする制御記憶の障害回避方法。

【請求項2】 請求項1記載の制御記憶の回避方法において、保守制御装置は、計算機システム稼働中の制御記憶障害発生時、その不良部分を回避するように対応表中の物理アドレスと論理アドレスの対応を再配置し、該配置された論理アドレスと物理アドレスに基づいて、該当マイクロプログラムコードに付加された物理アドレスを再配置し、該再配置後の物理アドレスに従って当該マイクロプログラムコードを制御記憶に再ロードすることを特徴とする制御記憶の障害回避方法。

【請求項3】 請求項1、2記載の制御記憶の障害回避方法において、保守制御装置は、物理アドレスの再配置時、対応表の物理アドレスにより、マイクロプログラムの起動アドレスを登録したテーブルの該当エントリを書き換えることを特徴とする制御記憶の障害回避方法。

【請求項4】 請求項1、2記載の制御記憶の障害回避方法において、保守制御装置は、実行状況のトレース時、トレース情報のマイクロプログラムの物理アドレスを、対応表により論理アドレスに変換することを特徴とする制御記憶の障害回避方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、マイクロプログラム制御の計算機システムにおける制御記憶の障害部位を回避する方法に関する。

【0002】

【従来の技術】 現在マイクロプログラムにより制御される演算装置をもつ計算機システムでは、マイクロプログラムによる計算機システムの障害処理等の事情でマイクロプログラムの量、即ち制御記憶の容量は増大の傾向にあり、それだけ制御記憶自身の耐障害機能が求められている。

【0003】 従来、制御記憶内の固定障害の発生に対し

て、計算機システムに接続された保守・診断専用のサービスプロセッサ（SVP）に内蔵されたファイル機構、あるいは主記憶装置上のハードウェア専用領域（HSA）よりマイクロプログラムを再ロードする時、制御記憶の空白（リザーブ）領域に障害発生部位のアドレスを持つマイクロプログラムをロードし、その後の演算装置の動作時に、マイクロプログラムをデコードする機構により制御記憶の障害発生部位アドレスをリザーブ領域のアドレスをに変換する、又はマイクロプログラムのアドレスを通常の計算機命令語の様に基底アドレスレジスタを使用した相対アドレスによる再配置可能な構造にし、基底アドレスレジスタを変更する、等により障害発生部位の使用を回避する方法が一般に知られている。しかし、これらの方法では、マイクロプログラム実行時、必ずアドレス変換、あるいは基底アドレス加算のプロセスが必要となり、演算装置の性能が低下し、演算装置の構造が複雑となり、計算機システムのコストアップを招く欠点がある。

【0004】 一方、これらの欠点を回避するため、次のような機能により制御記憶の固定障害を抑止する方法がある。

1、制御記憶内の固定障害を検出すると演算装置の実行を一時中断し、その事を障害の起きたアドレス情報とともにSVPに通報する。

2、SVPは自身が持つマイクロプログラムのアドレス管理情報を使用して、障害の起きたアドレスを持つマイクロプログラムのワードをSVPが持つファイル装置内のマイクロプログラムソースファイルから検索し、そのワードを制御記憶内の空いた部分にアサインしてアドレスを付け替えるため、ソースファイルを書き換える。

3、その後、SVPはソースファイルをアセンブル（リンケージ）し、新たなマイクロプログラムのコード列を生成する。このコード列は障害の起きたアドレス部位はアサインされないように空白となっているため、制御記憶に再ロードしてもその影響を被ることはない。また、SVPはアドレス管理情報に制御記憶の障害の部分が使用不可であることを記録する。

4、新たなマイクロプログラムコードをロードし、その後演算装置の動作を再開する。

【0005】 なお、このような制御記憶の固定障害回避手法は、例えば特開平6-83613号公報や特開平6-187147号公報などに記載されている。

【0006】

【発明が解決しようとする課題】 上記方法は、演算装置に制御記憶の障害発生部位の使用を回避するための特別な機構を設ける必要がない利点があるが、以下に記すような多くの問題点があり、現実の計算機システムに適用するのは困難である

1、計算機システムのマイクロプログラムを保守する為、制御記憶上のマイクロプログラムのあるアドレスの

10

20

30

40

50

ワードにパッチを適用する場合、前記アドレス移動が行われている可能性がある為、パッチを適用することが不可能である。又、この様な計算機システムに新しいマイクロプログラムをインストールする場合、それが障害発生部位を使用することになると、システムダウンを起こす事が考えられる。これらを回避するためには、マイクロプログラム改訂対象の計算機システムのSVPが持つマイクロプログラムソースファイル及びアドレス管理情報を取り寄せ、そのソースファイルを管理情報に従い障害発生部を回避する様に改訂し、当該計算機システムのSVPに送り、前記した手順により再アセンブルしロードすることになるが、この様な手順が顧客サイトにある計算機システム毎に必要となり、従来の様な全ての計算機システムに対し共通のパッチ、又は新マイクロプログラムのインストールにより変更を施す場合と比べてマイクロプログラム保守が非常に煩雑になる。

【0007】2、マイクロプログラム制御の演算装置では、命令、割込等の処理は制御記憶の特定の固定番地から始まるマイクロプログラムにより処理を実行するのが一般的であるが、その固定番地が前記再配置により移動すると、そのマイクロプログラムは動作不可能となり、演算装置は処理を続行できなくなる。

【0008】3、計算機システムで障害解析のために、マイクロプログラムの実行状況を知りたい場合、ハードウェアのトレース機能によりマイクロプログラムのトレースを採取し解析したい場合がある。このとき前記アドレス移動が行われていると、その移動したマイクロプログラムのトレース情報は、マイクロプログラム設計者が持つ初期ソースファイル（計算機システム出荷時にコードを生成したソースファイル、即ち制御記憶障害による移動前の古いアドレスを持つ）による参照／照合が出来ないため、解析不可能であり、トレースを解析するためには、その計算機システムのSVPが持つソースファイル及びアドレス管理情報を取り寄せなければならず、保守／デバッグ手順は煩雑である。

【0009】4、計算機システムが演算装置を複数個持ち、それらが共通のSVPにより制御及び保守されているマルチプロセッサシステムである場合、特に各演算装置上の制御記憶が共通のマイクロプログラムソースファイルにより管理されている場合に、ある一つの演算装置の制御記憶障害で、SVP内のソースファイルに前記アドレス移動が適用された後、システム内の別演算装置の制御記憶にそのマイクロプログラムがロードされると、最初の演算装置の制御記憶の障害発生アドレスと同じアドレス部位は使用不可能にされてしまう、即ち計算機システム内の全演算装置の制御記憶に各演算処理装置個別に発生する制御記憶障害の総和分の使用不可能な領域が出来てしまう事になる。この様な制御記憶の無効領域の増大は、アドレス移動が不可能となったり、又、機能追加等の為にマイクロプログラムの更新（追加）に対応が

できなくなる可能性がある。また、この様な無効領域を見越して制御記憶を増やすことは、計算機システムのコストアップをもたらす事も考えられる。

【0010】自明な回避策として、各演算装置に対応してソースファイルを持ち、制御記憶障害によるアドレス移動は各々のソースファイルを基に遂行し、他の演算装置に影響を及ぼす事を避けることが考えられるが、この解決策では前記1で述べたのと同様の問題点により、各演算処理装置毎にマイクロプログラム管理を行わなければならない、マイクロプログラム保守は更に煩雑になる。

【0011】5、計算機システム内で、アドレス移動に必要な制御記憶の空き領域が存在しない場合、制御記憶に障害が起こるとシステムダウンとなるが、事前にそれを知る手段を持たないため、保守が不可能である。

【0012】本発明の目的は、マイクロプログラム制御の計算機システムにおいて、制御記憶の障害部位を回避する際の上記のような問題点を解決し、マイクロプログラムの保守操作性の向上を図ることにある。

【0013】

【課題を解決するための手段】本発明は、制御記憶にロードするために、SVP上のファイルに置かれるマイクロプログラムコードに、制御記憶上のアドレス（物理アドレス）とマイクロプログラムのソースコードが持つアドレス（論理アドレス）の対応を付加する。

【0014】マイクロプログラムを制御記憶に初期ロードする際、SVPは、制御記憶をテストし、その不良部位を回避するように物理アドレスと論理アドレスとの対応表を生成し、該対応表に基づいて、前記マイクロプログラムソースコードに付加された物理アドレスを再配置し、該再配置後の物理アドレスに従ってマイクロプログラムを制御記憶にロードする生成した対応表は、計算機システム起動後も、そのまま保持しておく。

【0015】計算機システム稼働中の制御記憶障害発生時、SVPは、その不良部分を回避するように対応表中の物理アドレスと論理アドレスの対応を再配置し、該配置された論理アドレスと物理アドレスに基づいて、該マイクロプログラムコードに付加された物理アドレスを再配置し、該再配置後の物理アドレスに従って当該マイクロプログラムコードを制御記憶に再ロードする。

【0016】また、物理アドレスの再配置時、SVPは、対応表の物理アドレスにより、マイクロプログラムの起動アドレスを登録したテーブルの該当エントリを書き換える。

【0017】さらに、計算機システム実行状況のトレース時SVPは、トレース情報のマイクロプログラムの物理アドレスを、対応表により論理アドレスに変換して出力する。

【0018】なお、マイクロプログラムコードを生成するアセンブル手段（またはリンクエディット手段）にお

いて、マイクロプログラムコード及びパッチ作成時にマイクロプログラムコードとパッチに論理アドレスを付加するようにする。これにより、顧客サイト毎に存在する計算機システムに対して共通のマイクロプログラムコード及びパッチを適用することができる。

【0019】また、マルチプロセッサシステムである場合、各演算装置対応にアドレス対応表を持たせることにより、一個の演算装置で発生した制御記憶の障害をその演算装置固有のアドレス対応表により、その演算装置のみのマイクロプログラムの再配置を行うことで、他の演算装置の制御記憶の同一アドレス部位が使用できなくなることを防ぎ、制御記憶の無効領域が生じないようにできる。

【0020】さらには、制御記憶の障害による、マイクロプログラムの再配置の履歴をSVPのファイルに記録し、保守操作員の入力するコマンドによりSVPのコンソール装置にその履歴を表示する機構、あるいはSVPがその履歴及び前記アドレス対応表を監視し、制御記憶の空き領域が十分でない場合、その事を自動的に警告メッセージとして表示するようにする。

【0021】

【発明の実施の形態】以下、本発明の実施の形態について図面により具体的に説明する。

【0022】図1は本発明に係わる一実施の形態の全体構成図である。計算機システム100は演算装置1、それに接続された主記憶装置120、及び、これらの装置の保守を司る保守制御装置であるSVP150により構成されている。マルチプロセッサシステムの場合、演算装置2が構成に含まれる場合もある。なお、図1では演算装置が3個以上の場合は図示していないが、同様の演算装置が主記憶装置に接続可能である。演算装置1には、マイクロプログラムを格納する制御記憶115、演算装置1の実行状況をトレースするトレース部112、マイクロプログラムの起動アドレスを登録した起動テーブル114をそれぞれ含む。演算装置2の構成も同様である。主記憶装置120上には、ハードウェア専用領域であるHSA125が存在し、HSA181とSVP150間にはデータ転送バス181を具備する。又、制御記憶115、146へのアクセスバス180、182により、演算装置1、2あるいはSVP150による制御で制御記憶115、146にデータを転送する事が出来る。SVP150は、それ自身が持つ記憶装置にはマイクロプログラムコード155、各演算装置1、2に対応したアドレス変換テーブル（物理アドレスと論理アドレスの対応表）151、152が有り、さらに、制御記憶115、146にマイクロプログラムコードを転送する一時記憶であるバッファエリア153、アクセスバス180、182を使用し制御記憶115、146の障害を検出する制御記憶障害検出機構157を具備する。又、SVP150上のテーブルのバックアップを保持するフ

ァイル装置160、SVP150を通じて計算機システム100を操作するためのコンソール装置170が接続されている。

【0023】なお、SVP150の記憶装置上にある変換テーブル151、152、マイクロプログラムコード55、及びバッファエリア153は、主記憶装置120のHSA125上に置く事もできる。この場合、演算装置により変換テーブルの更新を行う事が可能である。ただし、これはマルチプロセッサシステムで1個の演算装置の障害発生によるテーブル操作を他の演算装置で実行する場合に限られる。

【0024】以上の計算機システム100は顧客サイトに設置されるが、以下の設備はマイクロプログラム設計者側に1システム有ればよい。

【0025】アセンブル手段20はそれ自体計算機であり、それが内蔵するファイル装置にはマイクロプログラムソースコード21を持っている。マイクロプログラム設計者は、該アセンブル手段20に接続されたコンソール装置であるマイクロプログラム変更手段10を用いてソースコードを変更した場合、アセンブル手段20に装備されたアセンブルプログラム及びリンケージプログラム25によりマイクロプログラムコード22を作成することができ、又、論理処理部24により新旧のコード22を比較しパッチデータ23を作成できる。マイクロプログラム設計者は、これらのマイクロプログラムコード22、パッチデータ23をコード転送手段30により顧客サイトに設置された各計算機システムに送る事ができる。なお、コード転送手段30はローカルエリアネットワーク、公衆回線による転送、あるいはフロッピーディスク、磁気テープ等による媒体の搬送の何れかにより実現される。

【0026】図2は、前記SVP150上のマイクロプログラムコード155、アドレス変換テーブル151、152の構成例を詳細に記述したものである。

【0027】変換テーブル200はエントリが上から制御記憶の物理アドレス0より始まる順列を構成し、各エントリには、有効フラグ201、マイクロプログラムソースコードが持つ論理アドレス202からなり、これにより物理アドレスから論理アドレスの変換を行う事が出来る。有効フラグ201はその物理アドレスが使用されているかを表し、制御記憶の空き領域を検索するために使用される。前述したように、この変換テーブルは各演算装置に対応して1個存在する。例えば200は図1の演算装置110のアドレス変換テーブル151に対応し、220は別の演算装置140のアドレス変換テーブル152に対応する。

【0028】マイクロプログラムコード300は図1の同マイクロプログラムコード155に対応し、マイクロコードテーブル310、クロスアレンステーブル330よりなる。マイクロコードテーブル310は、各1行

のエントリがマイクロプログラムの1ワードに対応し、上からマイクロプログラムソースコードの論理アドレス0より始まる順列を構成する。各エントリはそれぞれ、物理アドレス313、分岐先論理アドレス314、オブジェクトコード315、フラグ316、及びクロスリファレンステーブル330を指すポインタ317にフィールド分割されている。物理アドレス313には、340で示すに各演算装置に対応した物理アドレスを登録できる。同様にオブジェクトコード315の一部である分岐アドレス（論理アドレス314で指される分岐先ワードの物理アドレス）も、350で示す様に、各演算装置に対応した物理アドレスを持つことが出来る。フラグ316はアドレス再配置時に使用される。クロスリファレンステーブル330は分岐による参照を遡るために使用されるテーブルである。例えば、ポインタ317が指し示すエントリ332は、そのワードに分岐するワードの論理アドレスであり、次のポインタ334はそのワードに分岐する別のワードが存在した場合、それらワードのポインタを該テーブル330上に登録しチェイニングする為のものである。

【0029】図3に、テーブル310、330上のマイクロプログラムワード間の相互関係を示す。今、マイクロプログラムソースコード上でワードA、及びワードBからワードCへ分岐しているとする。この場合、テーブル310のワードCのエントリ上のポインタ317が指すテーブル330のエントリ332には、テーブル310上のワードAのエントリの論理アドレスが登録され、更にポインタ334が指すエントリにはテーブル313上のワードBの論理アドレスが登録される。図3の例では、これ以上ワードCに分岐するワードが無い場合、ワードBの論理アドレスが登録されるエントリ332に対応するポインタ334には0が登録される。これらのテーブル310、330は、初期状態でのテーブル間の参照関係がアセンブル（及びリンケージ）時に決定される。クロスリファレンステーブル330は、あるワードのアドレスが変更されたとき、影響を受ける他のワードを特定するために使用される。これについては後述する。

【0030】図2に戻り、アセンブル手段（図1の20）によりマイクロプログラムコードが作成された時点では、マイクロプログラムコード300一式が生成され、転送手段30により計算機システム100のSVP150に転送される。その時のアドレス対応は論理アドレス＝物理アドレスで各エントリが登録され、特に図2の340、350の物理アドレスは全て論理アドレスが初期値として設定される。これが、後述するように、アドレス再配置により対応する物理アドレスに書き換えられる。

【0031】パッチデータ500は、計算機システム100のマイクロプログラムアップデート時にアセンブル

手段20より送られるマイクロコードデータの1ワード分のフォーマットであり、必要に応じたワード数のデータをSVP150が受け取れる。パッチデータ500は、アップデート対象のワードの論理アドレス502、そのワードから分岐する先のワードを示す論理アドレス503、マイクロコード本体のオブジェクトコード504、このデータの属性を示すフラグ505からなり、フラグ505はそのワードが追加／削除／上書きのうち何れかであることを示す。

10 【0032】マイクロコードバッファエリア600は、図1の同バッファエリア153の構造を詳細に記述したものである。先にも述べたように、マイクロコードバッファエリア600はアドレス再配置又はマイクロプログラムのアップデート時に一時的に使用されるエリアであり、SVP150或いは主記憶装置120上のHSA125に置かれて制御記憶115、146にロードされるべきデータ等をバッファリングするもので、アップデートするマイクロプログラムのワードの物理アドレス601、そのエントリをロードするか否かを示す有効フラグ602、ロードするオブジェクトコード603からなる。なお、オブジェクトコードは分岐アドレスを含むが、後に説明するようにアドレス再配置による物理アドレスの変更は解決された状態でロードされる。

【0033】図4は、図1の計算機システムにおけるマイクロプログラム初期ロードのフローチャートである。以下、図2および図4を参照してマイクロプログラム初期ロード時の動作を説明する。

【0034】マイクロプログラム初期ロードは、計算機システム100を起動する時、SVP150のマイクロプログラムコード155（図2の300）を、各演算装置110、140内の制御記憶115、146にロードすることで行われる。この時点では、マイクロプログラムコードには再配置は行われておらず、アセンブル手段20によって設定された論理アドレス＝物理アドレスの状態、図2の変換テーブル200、220及びマイクロプログラムコード300が存在する。

【0035】マイクロプログラム初期ロードは、SVP150が演算装置を選択し（ステップ50）、各演算装置毎にステップ51～67の処理を行うことで達成される。ここでは演算装置1を例として扱う。制御記憶障害検出機構157により、制御記憶115のチェックを行う（ステップ51、52）。これは制御記憶115に書き込み／読み出し動作を行い、その部分が正常に機能する事を検査するもので、検査する単位は制御記憶のRAMの構成により異なるが、ここでは1ワード単位とする。制御記憶115に障害が無い場合、物理アドレス＝論理アドレスとして、マイクロコードテーブル310の該当エントリには触らない（ステップ53）。一方、障害を発見した場合、該当する物理アドレスを持つマイクロプログラムのワードの再配置を行う必要がある。そこ

で、変換テーブル（アドレス対応表）200の有効フラグ201を検索し、空白部を検索する（ステップ54、55）。必要な空白部が存在しない場合、コンソール装置170にエラーを表示し、マイクロプログラム初期ロードを異常終了させる（ステップ56、57）。空白部が存在する場合、アドレス移動すべきワードのマイクロコードテーブル310上の該当エントリ内の物理アドレス340（演算装置に対応するエントリ）を空白部のアドレスに書き換え（ステップ58）、変換テーブル200の該空白部（再配置用エントリ）に該当ワードの論理アドレスを登録し、その有効フラグ201を1とすることにより、そのエントリが空白でないことを登録する（ステップ60）。

【0036】次に、マイクロコードテーブル310の該当エントリのポインタ317が指し示すクロスリファレンステーブル330の参照ワード論理アドレス332、ポインタ334を用いて、マイクロコードテーブル310の該当エントリのコード内分岐アドレス350を書換える（ステップ61）。これを図5で簡単に説明する。今、制御記憶中のワードaがワードbに分岐している場合（A）、マイクロコードテーブル310のワードbエントリのポインタ317が指すクロスリファレンステーブル内の332にはワードaの論理アドレスが登録されている（C）。ここで、ワードbの物理アドレスY0部に障害が検出されると、ワードbに物理アドレスの移動（Y0→Y1）が行われるが（B）、それに伴いワードaの分岐アドレスを書き換えるため、マイクロコードテーブル310のワードbエントリのポインタ317が指すクロスリファレンステーブル330内の332に登録されたワードaの論理アドレスにより、それが指すマイクロコードテーブル310のワードaエントリの分岐アドレスを書き換える（C）。制御記憶の種類によっては分岐アドレスのみならず、パリティあるいはECCコードを書き換える必要がある。

【0037】図4に戻り、マイクロコードテーブル310の該当エントリのポインタ317が指し示すクロスリファレンステーブル330にチェーンされている別のエントリが存在するかを調べる（ステップ62）。エントリが存在すると（例えば図3）、そのワードに分岐する別のワードがあるので、チェーンされた全てのエントリわたってステップ61を繰り返す。

【0038】以上、ステップ51～62をその演算装置の制御記憶全てのワードが尽きるまで繰り返し、当該制御記憶115の全てのチェックを行なった後（ステップ64）、マイクロコードテーブル310上の全ワードのオブジェクトコード315を、それに物理アドレス340を付加してバッファエリア153に転送し、そのフラグ602を”有効”にする（ステップ65）。この時、オブジェクトコード内の分岐アドレス350及び付加する物理アドレス340は、図6の（D）に示すように、

障害発生部位を回避した物理アドレスに書き換えられている。

【0039】マイクロコードテーブル310上の全ワードをバッファエリア153に転送後、バッファエリア内のフラグ602が”有効”なエントリを、物理アドレス601に従い制御記憶115にロードする（ステップ66）。このロード終了後、フラグ602を”無効”にする（ステップ67）。

【0040】以上、ステップ50～67を演算装置毎に繰り返す。こうして、全ての演算装置の制御記憶にマイクロコードがロードされれば（ステップ68）、マイクロプログラム初期ロードは終了である。

【0041】図6は、計算機システム稼働中のアドレス再配置のフローチャートである。以下、図2および図6を参照して稼働時のアドレス再配置の動作を説明する。

【0042】システム稼働中に制御記憶障害が発生した場合、SVP150は障害が発生した制御記憶を持つ演算装置を一時停止し、その演算装置に対応する変換テーブル200を選択する（ステップ610）。障害検出機構157により制御記憶内の障害発生部のワードを選択する（ステップ611）。以下の再配置処理は図4のマイクロプログラム初期ロードとほぼ同じ手順で行う。すなわち、変換テーブル200を検索して制御記憶の空白部を探し（ステップ612）、変換テーブル200とマイクロコードテーブル310内の当該エントリの物理アドレスを書き換え（ステップ617、618）、更にクロスリファレンステーブル330より分岐アドレスの書き換えを行い（ステップ619、620）、バッファエリア600への転送（ステップ622）、制御記憶への転送を行い（ステップ623、624）、その後、演算装置の動作を再開するまた、この再配置処理で制御記憶に必要な空白部が存在しなくなった場合、コンソール装置170に警告を表示する（ステップ625、626）。初期ロードと異なるのは、障害発生演算装置と制御記憶の障害発生部位のワードが最初から認識されているため、処理対象が障害発生演算装置かつその制御記憶内の障害発生ワードに限定されることである。

【0043】次に、図1の計算機システムでのパッチによるマイクロコードの改訂の方法を説明する。アセンブル手段20にて、マイクロプログラムソースコード21をアセンブルプログラム及びリンケージプログラム25によりマイクロプログラムコード22に変換した後、論理処理部24により新旧コードの差分情報を作成し、それを（a）旧コードからの削除、（b）新規追加、（c）旧コードの変更の順番に抽出する。この差分情報に、論理アドレス、分岐先論理アドレス、及び削除／新規追加／旧コードの変更の種別をフラグとしたデータを付加し、図2の500に記したフォーマットのパッチデータを作成する。これを転送手段30を使用して計算機システム100に転送する。

【0044】 計算機システム100は受け取ったパッチデータをファイル装置160に一時蓄積し、その後、このパッチデータに従って、マイクロプログラムコード155、変換テーブル151、152を改訂する。これを図2及び図7のフローチャートを参照して説明する。

【0045】 計算機システム100は、パッチデータ500のフラグを参照してデータを削除、新規、変更の順に整列する(ステップ701)。データが削除の場合

(ステップ702)、パッチデータ500の論理アドレス502が示すマイクロコードテーブル310中の該当エントリを見つけ、その物理アドレス313が示す変換テーブル200中のエントリ202の内容をクリアし、フラグ201を”空白”にする(ステップ703)。その後、パッチデータ500の論理アドレス502が示すマイクロコードテーブル310中の当該エントリの内容313、314、315、316、317をクリアする(ステップ704)。以下、ステップ703、704を削除データが尽きるまで繰り返す(ステップ705)。次に、データが新規の場合(ステップ706)、パッチデータ500の論理アドレス502が示すマイクロコードテーブル310の該当空きエントリに該パッチデータの内容を入力し(ステップ707)、これを新規データ全てを処理するまで繰り返す(ステップ708)。また、データが変更の場合(ステップ710)、そのワードが分岐先論理アドレスの変更を含むとき分岐アドレス314に503の内容を入力し(ステップ712)、オブジェクトコード315に504の内容を入力し(ステップ713)、これを変更データ全てを処理するまで繰り返す(ステップ714)。

【0046】 ここで、分岐アドレスの変更が起きた場合、ワードの相互参照関係が変化するので、クロスリファレンステーブル330を再構築する必要がある。これを、図8のフローチャート及び図10の再構築前と再構築後のテーブルを参照して説明する。

【0047】 最初に、マイクロコードテーブル310の全エントリにあるポインタ317を初期化する(ステップ801)。次に、該マイクロコードテーブル310のエントリを指すポインタ*i*、クロスリファレンステーブル330のエントリを指すポインタ*j*を先頭に初期設定し(ステップ802)、ポインタ*i*が指すマイクロコードテーブル310のエントリに分岐先論理アドレス314(図10ではBA)が示すエントリのポインタ317の値(図10ではX)を、ポインタ*j*が示すクロスリファレンステーブル330のエントリ334に入力し、ポインタ*i*の値を332に入力した後、317にポインタ*j*の値を入力する(ステップ803)。その後、ポインタ*i*、*j*を、テーブル310、330各々の次のエントリを指すように更新し(ステップ804、805)、再びステップ803を実行する。以降、ステップ803ー

805をマイクロコードテーブル310のすべてのエントリが終わるまで繰り返す(ステップ806)。それが終了した時、図10の(A)は、同(B)で示す様な各ワード間の前後参照関係ができています。あるワードに分岐するワードが複数個あった場合、クロスリファレンステーブル330にはポインタ334でチェインされる。

【0048】 以上でパッチの前段階の処理が終わり、実際のパッチロード処理を行なう。これを図9で説明する。

【0049】 計算機システム内の一つの演算装置1を選択する(ステップ901)。以下、ステップ901ー918の処理が各々の演算装置毎に繰り返される。まず、マイクロコードテーブル310のエントリの中で、先のステップ806までの処理で分岐先が変更されたワードを抽出する(ステップ902)。これは、ここまでの処理でフラグ316にそのことを登録すれば簡単に認識できる。次に、その変更された分岐先の物理アドレスを、マイクロコードテーブル310の物理アドレス340の該当演算装置番号に対応したエントリから取り出し、自ワードの分岐アドレス350のやはり該当演算装置番号に対応したエントリに格納する(ステップ903)。分岐先が変更された全てのワードに関して、ステップ903の処理を繰り返す(ステップ904)。

【0050】 次に、ステップ902と同様な手段でパッチで空きエリアに入る新規に作られたワードを抽出する(ステップ905)。新たに使用する制御記憶115のチェックを行なう為、障害検出機構157により制御記憶115の該当部分を検査し(ステップ906)、障害がなければ、物理アドレス=論理アドレスとする(ステップ911)。障害を発見した場合は、アドレス変換テーブル200の有効フラグ201を検索し、空白部を調べる(ステップ907)。必要な空白部が存在しない場合、コンソール装置170にエラーを表示し(ステップ908)、パッチのロードを異常終了させる(ステップ909)。空白部が存在する場合、物理アドレス=空白部アドレスとする(ステップ910)。ここでは物理アドレスと論理アドレスが異なる為、このワードに分岐するワードの分岐アドレス350を書き換える必要が有る。そのため、その様なワードをクロスリファレンステーブル330から検索し、そのワードの分岐アドレス350の演算装置に対応したエントリに物理アドレスを入力する(ステップ910)。更に、物理アドレスが示す変換テーブル200のエントリ202に元の論理アドレスを登録し、フラグ201を1とすることにより、そのエントリが空白でない事を登録し、変換テーブル200を改定する(ステップ912)。ステップ906から912を、新規に作られたワードが無くなるまで繰り返す(ステップ913)。

【0051】 以上の処理が終われば、マイクロコードテーブル310から今までの処理で変更の有った全ワード

10

20

30

40

50

の物理アドレス313、オブジェクトコード315に分岐先アドレス350をマージしたデータをバッファエリア600(図1の153)に転送する(ステップ914)。以下、アドレス再配置の場合と同様に演算装置を一時停止し(ステップ915)、バッファエリア600より制御記憶115(又は116)にデータをロードする(ステップ916)。その後、演算装置を再起動し(ステップ917)、当該演算装置へのパッチは終了する。全ての演算装置に関してステップ901-917を繰り返して(ステップ918)、計算機システム100へのパッチがすべて終了する。

【0052】次に、固定番地から始まるマイクロプログラムの先頭固定番地が再配置された時の命令起動処理について説明する。図11は、演算装置内の起動テーブル(図1の114、144)の働きを説明する図である。起動テーブル1004の各エントリは演算装置により命令とそれのマイクロプログラムのスタートアドレスが一对一に対応がとられており、命令パイプラインにおいて、命令フェッチ(ステップ1000、命令デコード(ステップ1002)と進み、該命令デコード時に演算装置により起動テーブル1004中の該当エントリが選択され、その先頭アドレスが読み出され(ステップ1005)、命令実行時に(ステップ1006)、そのアドレスで制御記憶のマイクロプログラムが起動される(ステップ1007)。この起動テーブル1004の先頭アドレスは、SVP150により書き換えることができる。

【0053】先のアドレス移動とマイクロプログラムの初期ロード及びパッチにより先頭アドレスが再配置された時(図4のステップ60、図6のステップ618、図9のステップ912で起動テーブルに登録されているアドレスが再配置された時)、SVP150は起動テーブル1004(図1の114、144)に新しい物理アドレスを書き込む。これによりアドレスが再配置以降のマイクロプログラム起動も動作が保証される。

【0054】次に、図1の計算機システム100でのマイクロプログラムトレース採取法を説明する。トレース部112、142が演算装置110、140のトレースデータを採取した時、通常、そのデータはSVP150に取り出される。その時、変換テーブル151、152(図2の200、220)を取り出し、トレースデータと共に保守担当者によりコード転送手段30を使いマイクロプログラム設計者に届けられる。トレースデータはマイクロプログラムが実行された順の制御記憶115、146の物理アドレスの順番からなっており、これを変換テーブル200、220を使用して論理アドレスに変換する。あるいは、トレースを採取した時点で、SVP150が変換テーブル200、220を参照し、トレースデータを論理アドレスに変換してトレースデータを得る事もできる。これにより、マイクロプログラム設計者

は、自ら設計したソースコードと整合したトレースデータを得る事ができる。

【0055】以上説明した実施形態において、マイクロプログラム再配置及びパッチによるマイクロコードの改訂処理では、変換テーブル200及びマイクロコードテーブル310内の演算装置対応の物理アドレス340により、再配置による物理アドレス-論理アドレスの対応の変更が、その演算装置に局所化される。これにより、マルチプロセッサシステムの場合、アドレス再配置は障害を起こした演算装置以外には波及せず、それら演算装置の制御記憶には無効領域が生じない。

【0056】又、図4のステップ55-57、図6のステップ613-614、図9のステップ907-909をみればわかる様に、制御記憶の空き領域が十分でない、即ち、アドレス移動、マイクロプログラムの初期ロード、又はパッチに必要な量が存在しない場合、その事を自動的に警告メッセージとして表示している。これにより、保守操作員はシステムダウンを予想し、計算機システム保守の契機を知ることができる。

【0057】

【発明の効果】本発明の制御記憶の障害回避方法によれば、以下のような効果が得られる。

(1) 顧客サイト毎に存在する計算機システムに対して、論理アドレスをキーに共通のマイクロプログラムコード及びパッチを適用することができ、マイクロプログラムの保守操作性が向上する。

(2) マイクロプログラム再配置時に、移動先の物理アドレスを、演算装置に設けられた起動アドレスを登録したテーブルのエントリに上書きすることにより、アドレスの再配置が行われたマイクロプログラムを持つ演算装置の実行を保証できる。

【0058】(3) 計算機システム内演算装置の実行状況をトレースする際、トレース情報の物理アドレスを論理アドレスに変換して出力することにより、該出力されたトレース情報と論理アドレスをキーとしたマイクロプログラムのソースリストとのアドレス整合性を確保することができ、任意の計算機システムで採取されたトレース情報を共通のソースリストにより解析できる。

【図面の簡単な説明】

【図1】本発明における計算機システムの制御記憶のアドレス再配置を行なう全体的機構を示した図である。

【図2】図1のSVP上のアドレス再配置に関するテーブル類の詳細構成例を示した図である。

【図3】テーブル上のマイクロプログラムワード間の相互関係を示した図である。

【図4】本発明における計算機システムの制御記憶のマイクロプログラム初期ロードを示したフローチャートである。

【図5】制御記憶のアドレス再配置を行なう時の制御記憶とテーブルの取扱いを示した図である。

【図6】制御記憶のアドレス再配置のフローチャートである。

【図7】計算機システムにパッチを適用する時のフローチャートの一部である。

【図8】計算機システムにパッチを適用する時のフローチャートの続きである。

【図9】計算機システムにパッチを適用する時のフローチャートの更に続きである。

【図10】クロスリファレンステーブルの再構築を説明する図である。

【図11】起動テーブルの働きを示した図である。

【符号の説明】

- 10 マイクロプログラム変更手段
- 20 アセンブル（リンケージ）手段
- 21 マイクロプログラムソースコード
- 22 マイクロプログラムコード
- 23 マイクロプログラムパッチデータ
- 24 新旧マイクロプログラムソースコードよりパッチデータを生成する手段

*

* 25 アセンブル&リンケージプログラム

30 マイクロプログラムコード&パッチデータ転送手段

100 計算機システム

110、140 計算機システム内の演算装置

112、142 マイクロプログラムトレース採取部

114、144 マイクロプログラム起動テーブル

115、146 制御記憶

120 主記憶装置

10 125 主記憶装置のハードウェア専用領域（HSA）

142 マイクロプログラムトレース採取装置

150 SVP

151、152 アドレス変換テーブル

153 マイクロプログラム改定用のバッファエリア

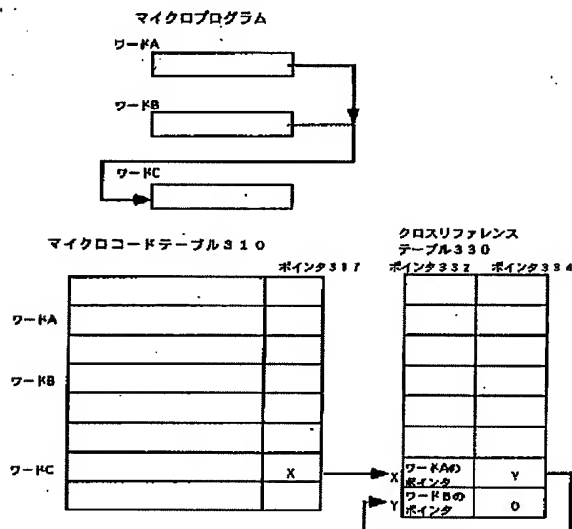
155 SVP150上のマイクロプログラムコード

157 制御記憶障害検出機構

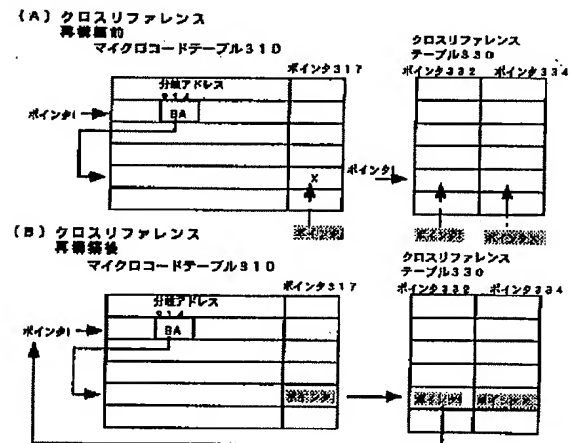
160 ファイル装置

170 コンソール装置

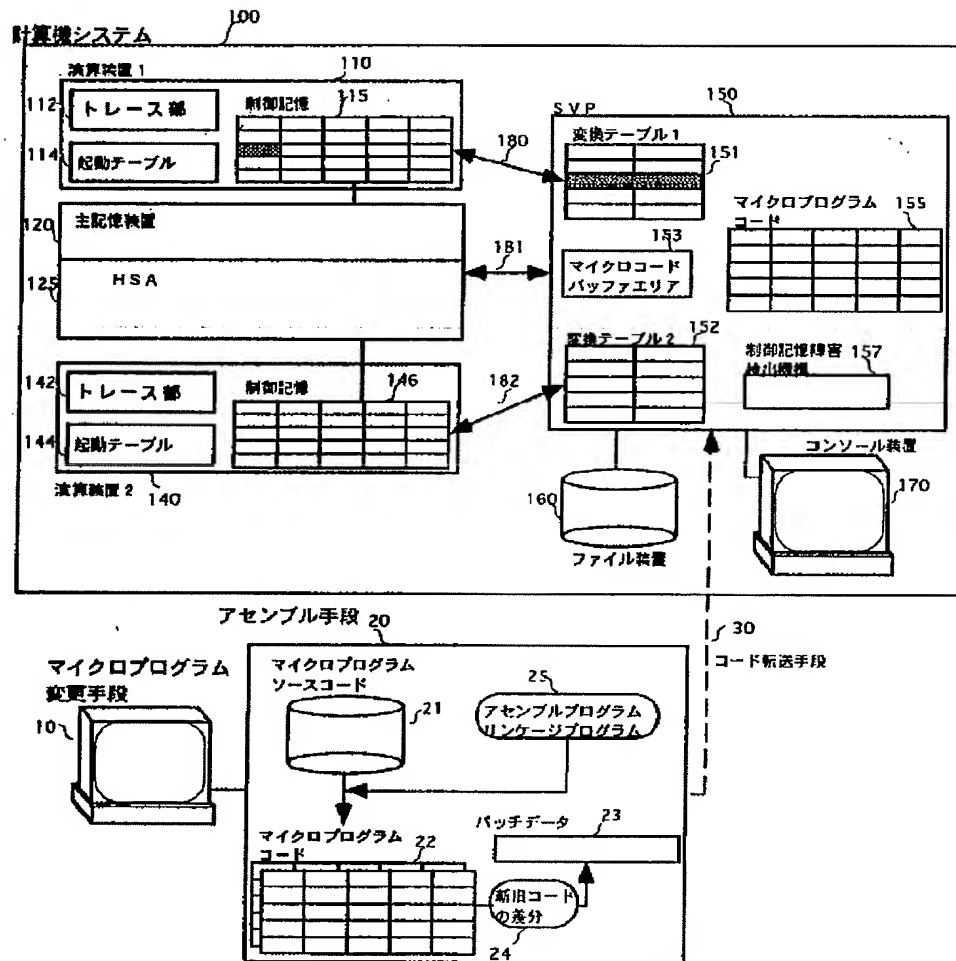
【図3】



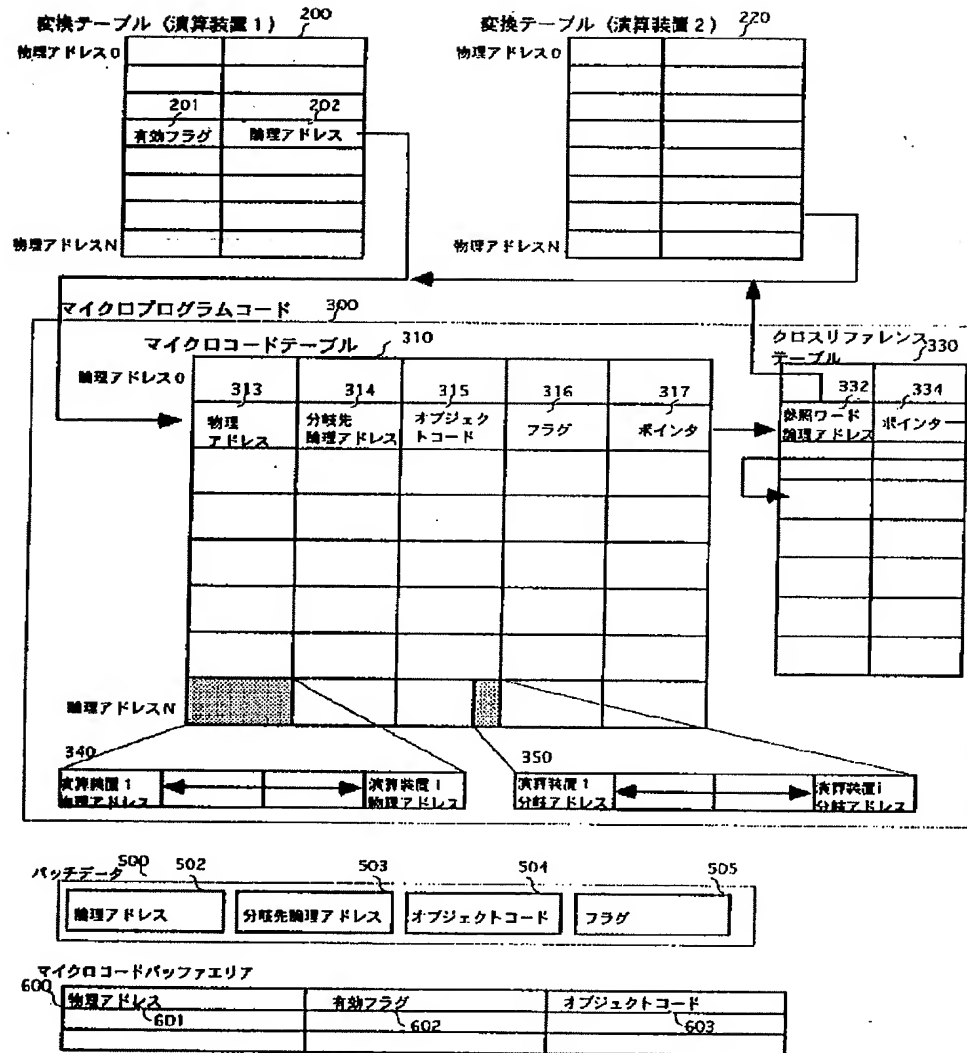
【図10】



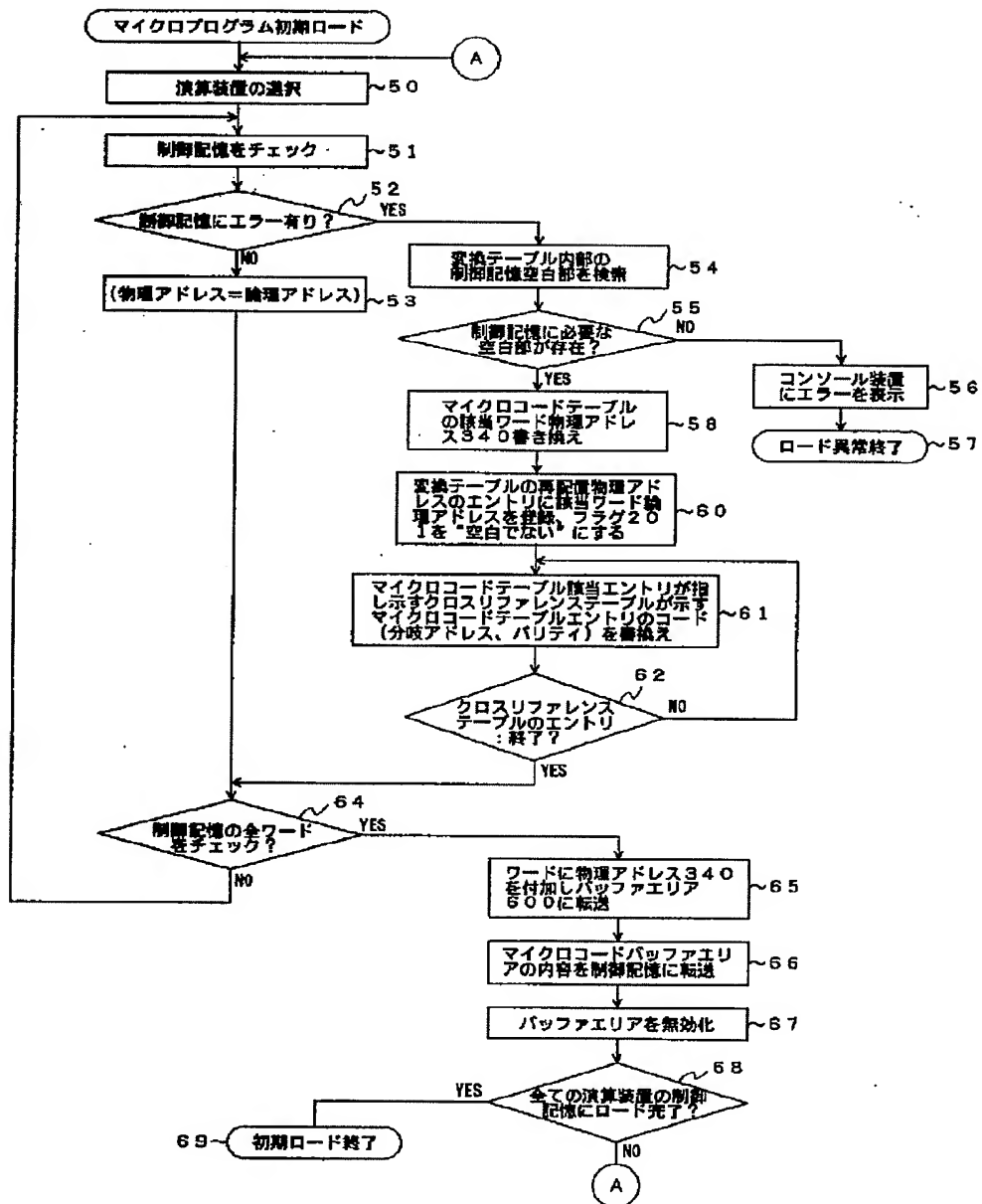
【図1】



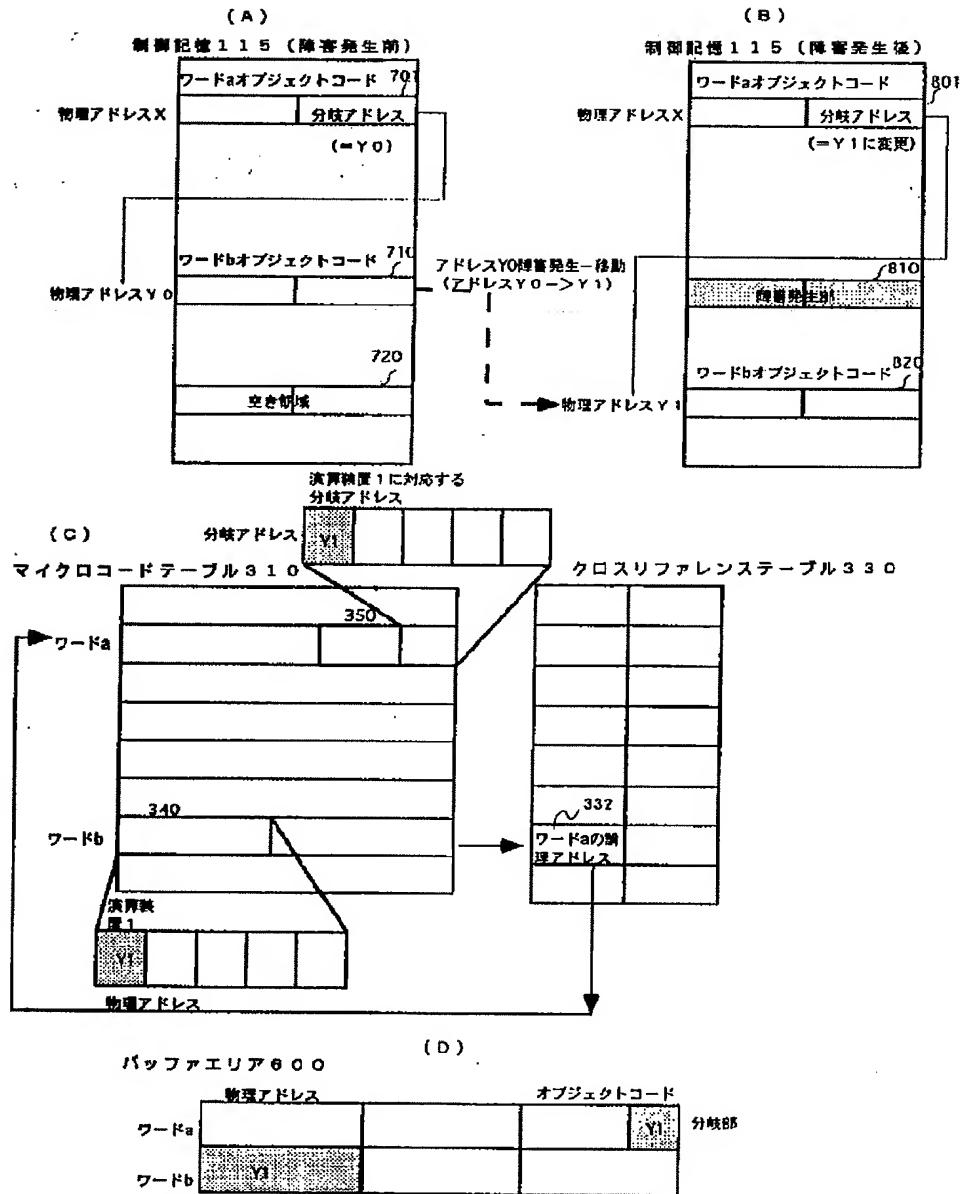
【図2】



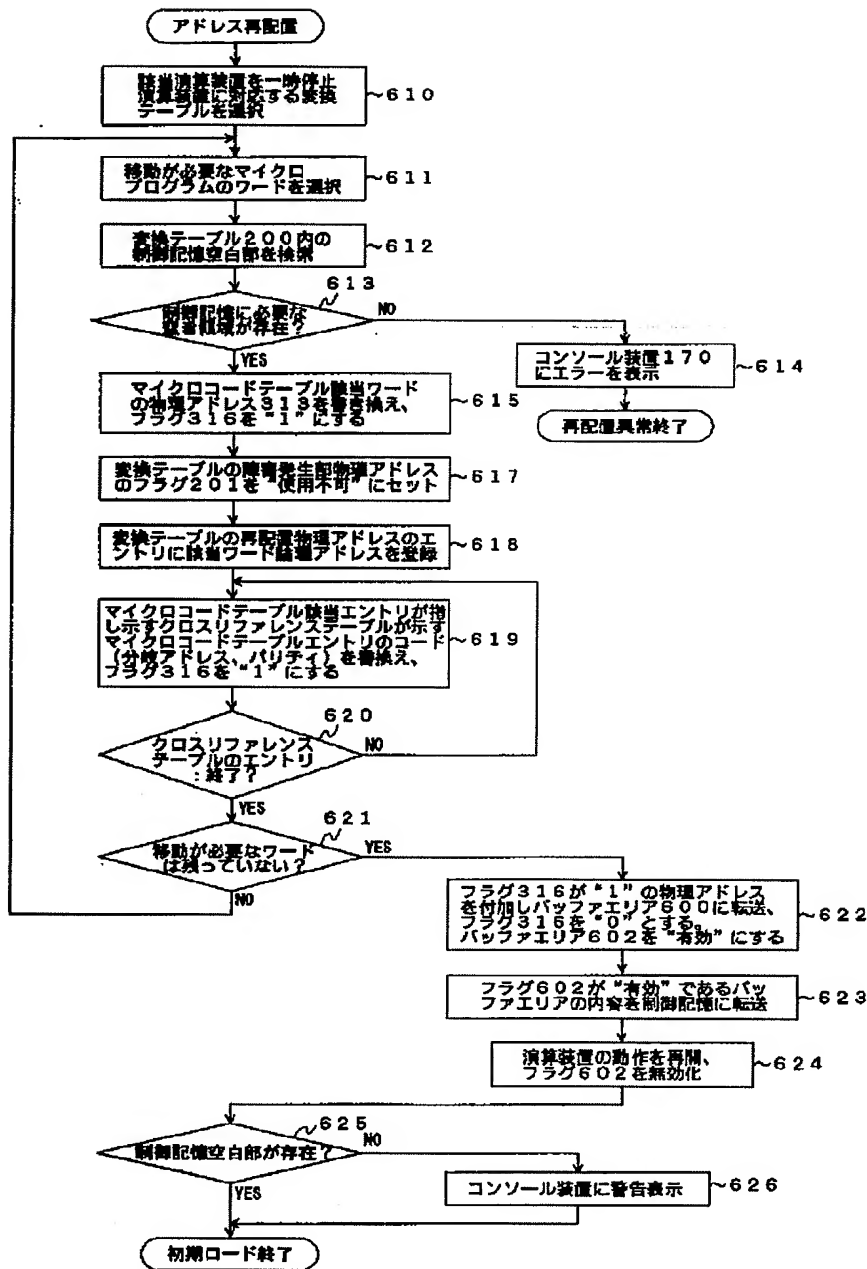
【図4】



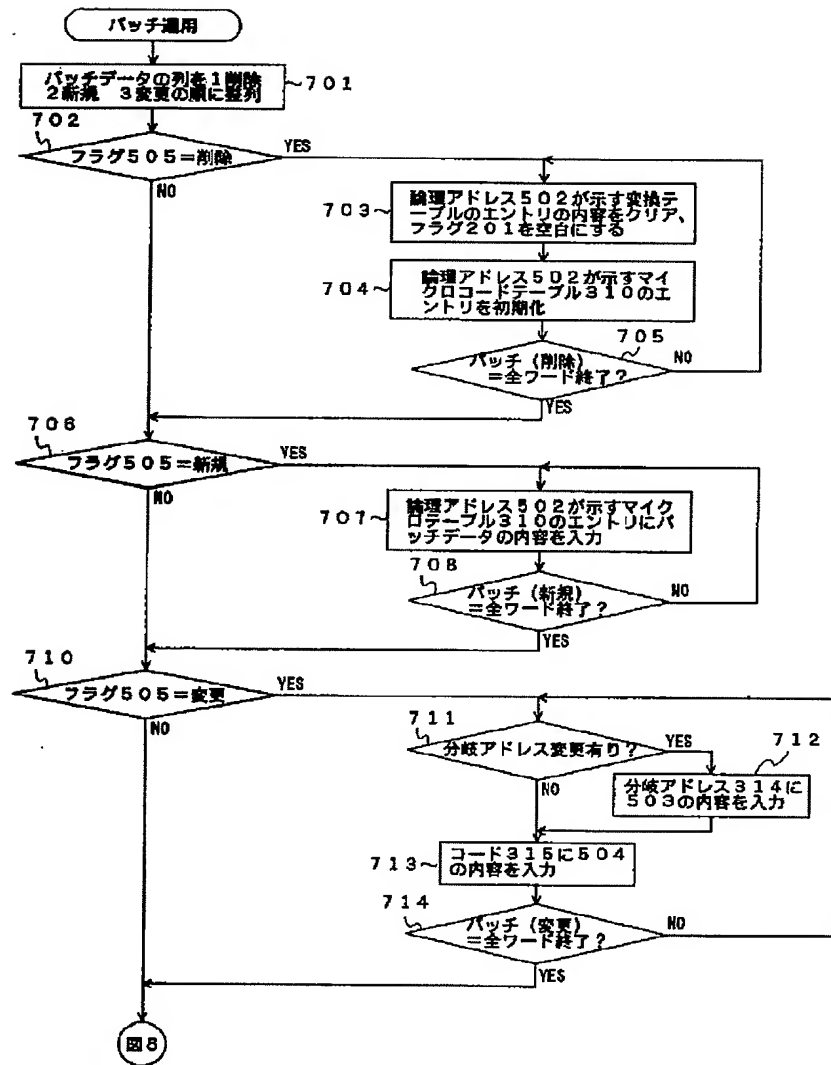
【図5】



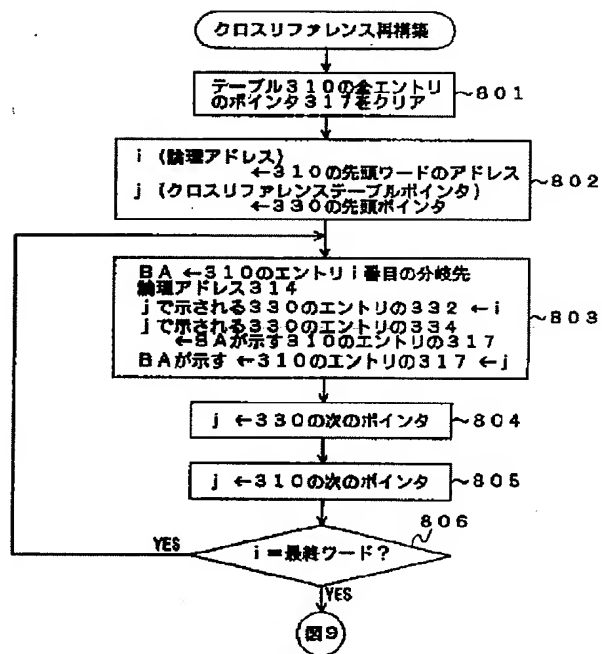
【図6】



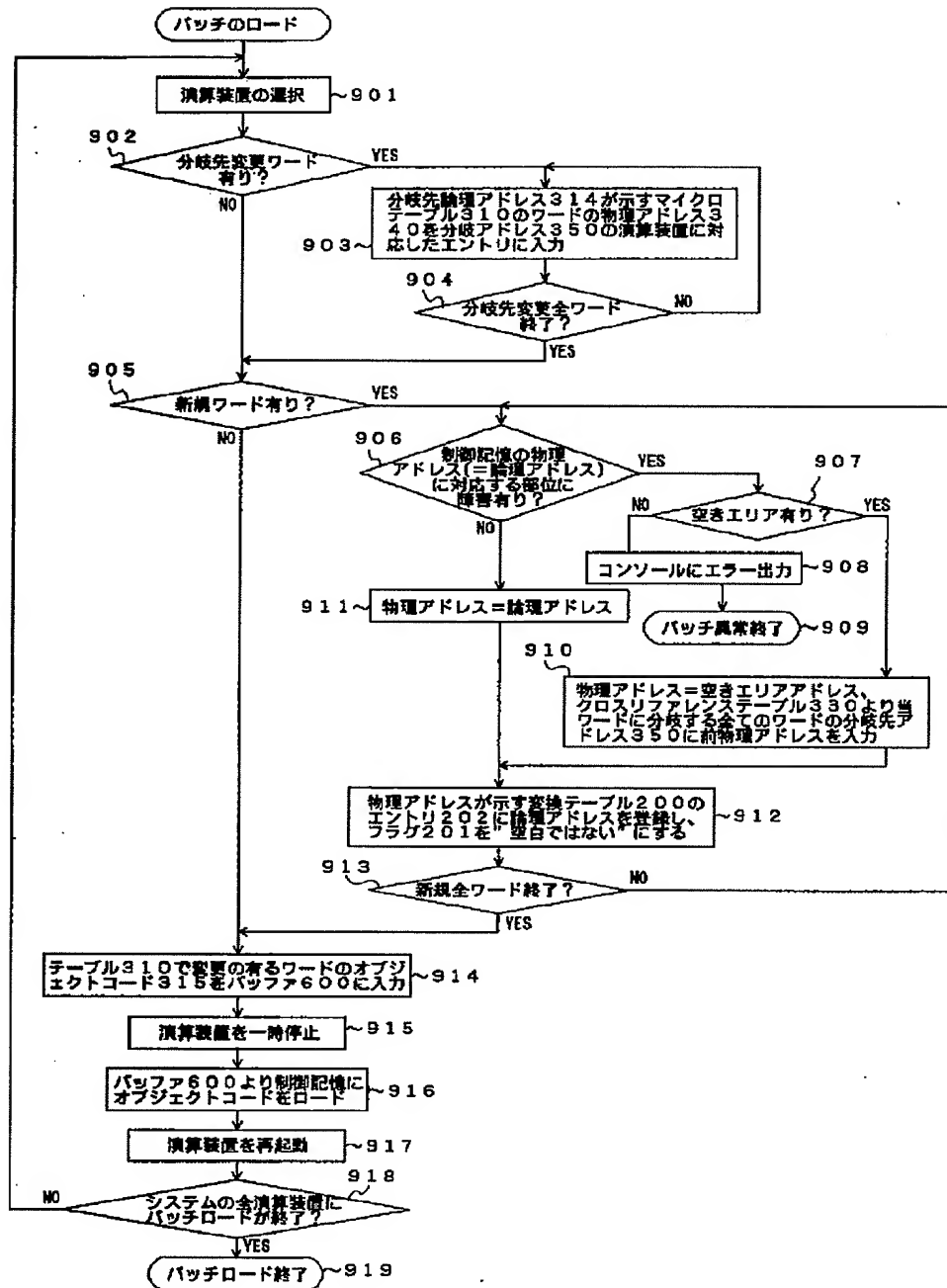
【図7】



【図8】



【図9】



【図11】

